quanteda

September 23, 2014

Type Package

Title Quantitative Analysis of Textual Data

Version 0.5.0

Date 2014-09-23

Author Ken Benoit and Paul Nulty

Maintainer Ken Benoit <kbenoit@lse.ac.uk> and Paul Nulty <p.nulty@lse.ac.uk>

Description A package for the management and quantitative analysis of textual data with R. quanteda makes it easy to manage texts in the form of a corpus, defined as a collection of texts that includes document-level variables specific to each text, as well as meta-data for documents and for the collection as a whole. quanteda includes tools to make it easy and fast to manuipulate the texts the texts in a corpus, for instance by tokenizing them, with or without stopwords or stemming, or to segment them by sentence or paragraph units. quanteda implements bootstrapping methods for texts that makes it easy to resample texts from pre-defined units, to facilitate computation of confidence intervals on textual statistics using techniques of non-parametric bootstrapping, but applied to the original texts as data. quanteda includes a suite of sophisticated tools to extract features of the texts into a quantitative matrix, where these features can be defined according to a dictionary or thesaurus, including the declaration of collocations to be treated as single features. Once converted into a quantitative matrix (known as a ``dfm" for document-feature matrix), the textual feature can be analyzed using quantitative methods for describing,comparing, or scaling texts, or used to train machine learning methods for class prediction.

Encoding UTF-8

License GPL-3

Requires SnowballC

Imports wordcloud,proxy

Suggests austin,entropy,jsonlite,openNLP,RJSONIO,RCurl,twitteR,XML,lda,topicmodels

URL http://github.com/kbenoit/quanteda

LazyData TRUE

R topics documented:

bigrams *Create bigrams*

Description

Create bigrams

Usage

```
bigrams(text, window = 1, concatenator = "_", include.unigrams = FALSE,
  ...)
```

Arguments

text	character vector containing the texts from which bigrams will be constructed
window	how many words to be counted for adjacency. Default is 1 for only immediately neighbouring words. This is only available for bigrams, not for ngrams.
concatenator	character for combining words, default is _ (underscore) character
include.unigrams	if TRUE, return unigrams as well
...	provides additional arguments passed to tokenize

Value

a character vector of bigrams

Author(s)

Ken Benoit and Kohei Watanabe

Examples

```
bigrams("The quick brown fox jumped over the lazy dog.")
bigrams(c("The quick brown fox", "jumped over the lazy dog."))
bigrams(c("The quick brown fox", "jumped over the lazy dog."), window=2)
```

changeunits *change the document units of a corpus*

Description

For a corpus, recast the documents down or up a level of aggregation. "Down" would mean going from documents to sentences, for instance. "Up" means from sentences back to documents. This makes it easy to reshape a corpus from a a collection of documents into a collection of sentences, for instance.

Usage

```
changeunits(corp, to = c("sentences", "paragraphs", "documents"), ...)
```

Arguments

corp	corpus whose document units will be reshaped
to	new documents units for the corpus to be recast in
...	passes additional arguments to segment

Examples

```
# simple example
mycorpus <- corpus(c(textone="This is a sentence.  Another sentence.  Yet another.",
                     textwo="Première phrase.  Deuxième phrase."),
                   docvars=list(country=c("UK", "USA"), year=c(1990, 2000)),
                   notes="This is a simple example to show how changeunits() works.")
language(mycorpus) <- c("english", "french")
summary(mycorpus)
summary(changeunits(mycorpus, to="sentences"), showmeta=TRUE)

# example with inaugural corpus speeches
mycorpus2 <- subset(inaugCorpus, Year>2004)
mycorpus2
paragCorpus <- changeunits(mycorpus2, to="paragraphs")
paragCorpus
summary(paragCorpus, 100, showmeta=TRUE)
## Note that Bush 2005 is recorded as a single paragraph because that text used a single
## \n to mark the end of a paragraph.
```

---

clean *simple cleaning of text before processing*

Description

clean removes punctuation and digits from text, using the regex character classes for punctuation and digits. clean uses the standard R function tolower to convert the text to lower case. Each of these steps is optional, but switched on by default, so for example, to remove punctuation and convert to lower, but keep digits, the command would be: clean(mytexts, removeDigits=FALSE)

Usage

```
clean(x, ...)

## S3 method for class character
clean(x, removeDigits = TRUE, removePunct = TRUE,
  lower = TRUE, ...)

## S3 method for class corpus
clean(x, removeDigits = TRUE, removePunct = TRUE,
  lower = TRUE, ...)
```

Arguments

x	The object to be cleaned. Can be either a character vector or a corpus object. If x is a corpus, clean returns a copy of the x with the texts cleaned.
...	additional parameters
removeDigits	remove numbers if TRUE
removePunct	remove punctuation if TRUE
lower	convert text to lower case TRUE

Value

A character vector equal in length to the original texts, after cleaning.

Examples

```
clean("This is 1 sentence with 1.9 numbers in it, and one comma.", removeDigits=FALSE)
clean("This is 1 sentence with 1.9 numbers in it, and one comma.", lower=FALSE)

# for a vector of texts
clean(c("This is 1 sentence with 1.9 numbers in it, and one comma.",
        "€1.2 billion was spent on text analysis in 2014."))
```

---

collocations *Detect collocations in a text*

Description

returns a list of collocations. Note: Currently works only for pairs (bigram collocations).

Usage

```
collocations(text = NULL, file = NULL, top = NA, distance = 2, n = 2,
  method = c("lr", "chi2", "mi"))
```

Arguments

text	a text or vector of texts
file	a filename containing a text
top	threshold number for number of collocations to be returned (in descending order of association value)
distance	distance between pairs of collocations
n	Only bigrams (n=2) implemented so far.
method	association measure for detecting collocations

Value

A list of collocations, their frequencies, and their test statistics

Author(s)

Kenneth Benoit

Examples

```
collocations(texts(inaugCorpus)[1], top=50)
collocations(texts(inaugCorpus)[1], top=50, method="chi2")
```

---

corpus *Constructor for corpus objects*

Description

Creates a corpus from a document source, such as character vector (of texts), or an object pointing to a source of texts such as a directory containing text files. Corpus-level meta-data can be specified at creation, containing (for example) citation information and notes.

Usage

```
corpus(x, ...)

## S3 method for class directory
corpus(x, enc = NULL, docnames = NULL,
  docvarsfrom = c("filenames", "headers"), docvarnames = NULL, sep = "_",
  source = NULL, notes = NULL, citation = NULL, ...)

## S3 method for class character
corpus(x, enc = NULL, docnames = NULL, docvars = NULL,
  source = NULL, notes = NULL, citation = NULL, ...)

is.corpus(x)
```

Arguments

x	A source of texts to form the documents in the corpus. This can be a filepath to a directory containing text documents (see directory), or a character vector of texts.
...	additional arguments
enc	A string (or character vector) specifying the encoding for each text in the corpus. Must be a valid entry in iconvtypes().
docnames	Names to be assigned to the texts, defaults to the names of the character vector (if any), otherwise assigns "text1", "text2", etc.
docvarsfrom	Argument to specify where docvars are to be taken, from parsing the filenames (filenames) separated by sep or from meta-data embedded in the text file header (headers).
docvarnames	Character vector of variable names for docvars
sep	Separator if docvar names are taken from the filenames.
source	A string specifying the source of the texts, used for referencing.
notes	A string containing notes about who created the text, warnings, To Dos, etc.
citation	Information on how to cite the corpus.
docvars	A data frame of attributes that is associated with each text.

Value

A corpus class object containing the original texts, document-level variables, document-level meta-data, corpus-level metadata, and default settings for subsequent processing of the corpus. A corpus consists of a list of elements described below, although these should only be accessed through accessor and replacement functions, not directly (since the internals may be subject to change). The structure of a corpus classed list object is:

`$documents`	A data frame containing the document level information, consisting of `texts`, user-named `docvars` variables describing attributes of the documents, and `metadoc` document-level metadata whose names begin with an underscore character, such as `_language`.
`$metadata`	A named list set of corpus-level meta-data, including `source` and `created` (both generated automatically unless assigned), `notes`, and `citation`.
`$settings`	Settings for the corpus which record options that govern the subsequent processing of the corpus when it is converted into a document-feature matrix (dfm). See settings.
`$tokens`	An indexed list of tokens and types tabulated by document, including information on positions. Not yet fully implemented.

`is.corpus` returns `TRUE` if the object is a corpus

See Also

docvars, metadoc, metacorpus, language, encoding, settings, texts

Examples

```
## Not run:
# import texts from a directory of files
corpus(directory("~/Dropbox/QUANTESS/corpora/ukManRenamed"),
       enc="UTF-8",
       source="Kens UK manifesto archive")

# choose a directory using a GUI
corpus(directory())
## End(Not run)
#
# create a corpus from texts
corpus(inaugTexts)

# create a corpus from texts and assign meta-data and document variables
uk2010immigCorpus <- corpus(uk2010immig,
                            docvars=data.frame(party=names(uk2010immig)),
                            enc="UTF-8")
```

`countSyllables` *Returns a count of the number of syllables in the input*

Description

This function takes a text and returns a count of the number of syllables it contains. For British English words, the syllable count is exact and looked up from the CMU pronunciation dictionary. For any word not in the dictionary the syllable count is estimated by counting vowel clusters.

Usage

```
countSyllables(sourceText)
```

Arguments

sourceText	Character vector of texts whose syllables will be counted

Details

This only works for English.

Value

numeric Named vector of counts of the number of syllables for each element of sourceText. When a word is not available in the lookup table, its syllables are estimated by counting the number of (English) vowels in the word.

Examples

```
countSyllables("This is an example sentence.")
myTexts <- c("Text one.", "Superduper text number two.", "One more for the road.")
names(myTexts) <- paste("myText", 1:3, sep="")
countSyllables(myTexts)
```

---

describeTexts *print a summary of texts*

Description

Prints to the console a desription of the texts, including number of types, tokens, and sentences

Usage

```
describeTexts(txts, verbose = TRUE)
```

Arguments

txts	The texts to be described
verbose	Default is TRUE. Set to false to suppress output messages

Examples

```
describeTexts(c("testing this text", "and this one"))
describeTexts(uk2010immig)
```

dfm *Create a document-feature matrix from a corpus object*

Description

Returns a document by feature matrix with additional meta-information (settings, identification of training texts for supervised models, resampling information, etc.) that is useful in other quanteda functions. A typical usage would be to produce a word-frequency matrix where the cells are counts of words by document, but the definition of "features" is entirely general.

Usage

```
dfm(x, ...)

## S3 method for class corpus
dfm(x, feature = c("word"), stem = FALSE,
  stopwords = NULL, bigram = FALSE, groups = NULL, verbose = TRUE,
  dictionary = NULL, dictionary_regex = FALSE, clean = TRUE,
  addto = NULL, ...)

## S3 method for class character
dfm(x, feature = c("word"), stem = FALSE,
  stopwords = NULL, bigram = FALSE, verbose = TRUE, dictionary = NULL,
  dictionary_regex = FALSE, clean = TRUE, addto = NULL, ...)

is.dfm(x)
```

Arguments

x	Corpus or character vector from which to generate the document-feature matrix
...	additional arguments passed to clean
feature	Feature to count (e.g. words)
stem	Stem the words
stopwords	A character vector of stopwords that will be removed from the text when constructing the dfm. If NULL (default) then no stopwords will be applied. If "TRUE" then it currently defaults to stopwords.
bigram	include bigrams as well as unigram features, if TRUE
groups	Grouping variable for aggregating documents
verbose	Get info to screen on the progress
dictionary	A list of character vector dictionary entries, including regular expressions (see examples)
dictionary_regex	TRUE means the dictionary is already in regular expression format, otherwise it will be converted from "wildcard" format
addto	NULL by default, but if an existing dfm object is specified, then the new dfm will be added to the one named. If both dfm's are built from dictionaries, the combined dfm will have its Non_Dictionary total adjusted.

Details

`is.dfm` returns `TRUE` if and only if its argument is a dfm.

Value

A specially classed matrix object with row names equal to the document names and column names equal to the feature labels. Additional information is attached to this object as `attributes`, such as settings.

Author(s)

Kenneth Benoit

Examples

```
data(inaugCorpus)
wfm <- dfm(inaugCorpus)

## by president, after 1960
wfmByPresfrom1900 <- dfm(subset(inaugCorpus, Year>1900), groups="President")
docnames(wfmByPresfrom1900)

## with dictionaries
data(inaugCorpus)
mycorpus <- subset(inaugCorpus, Year>1900)
mydict <- list(christmas=c("Christmas", "Santa", "holiday"),
               opposition=c("Opposition", "reject", "notincorpus"),
               taxing="taxing",
               taxation="taxation",
               taxregex="tax*")
dictDfm <- dfm(mycorpus, dictionary=mydict)
dictDfm

## removing stopwords
testText <- "The quick brown fox named Seamus jumps over the lazy dog also named Seamus, with
             the newspaper from a a boy named Seamus, in his mouth."
testCorpus <- corpus(testText)
settings(testCorpus, "stopwords")
dfm(testCorpus, stopwords=TRUE)
```

---

dfm2ldaformat *Convert a dfm into the format needed by lda*

Description

Convert a quanteda dfm object into the indexed format required by the topic modelling package lda.

Usage

```
dfm2ldaformat(d)
```

Arguments

d	A `dfm` object

Value

A list with components "documents" and "vocab" as needed by lda.collapsed.gibbs.sampler

Examples

```
mycorpus <- subset(inaugCorpus, Year>1970)
d <- dfm(mycorpus, stopwords=TRUE)
d <- trimdfm(d, minCount=5, minDoc=3)
td <- dfm2ldaformat(d)
if (require(lda)) {
    tmodel.lda <- lda.collapsed.gibbs.sampler(documents=td$documents,
                                              K=10,
                                              vocab=td$vocab,
                                              num.iterations=50, alpha=0.1, eta=0.1)
    top.topic.words(tmodel.lda$topics, 10, by.score=TRUE) # top five words in each topic
}
```

`dfm2tmformat` *Convert a dfm into a* **tm** *DocumentTermMatrix*

Description

tm represents sparse document-feature matrixes in the simple triplet matrix format of the package **slam**. This function converts a `dfm` into a `DocumentTermMatrix`, enabling a dfm to be used with other packages that expect this format, such as **topicmodels**.

Usage

```
dfm2tmformat(d, weighting = weightTf, ...)
```

Arguments

`d`	A dfm object
`weighting`	weight function arguments passed to `as.TermDocumentMatrix`, defaults to term frequency (see `as.DocumentTermMatrix` for a list of options, such as tf-idf).

Value

A simple triplet matrix of class as.DocumentTermMatrix

Examples

```
mycorpus <- subset(inaugCorpus, Year>1970)
d <- trimdfm(dfm(mycorpus), minCount=5, minDoc=3)
dim(d)
td <- dfm2tmformat(d)
length(td$v)
if (require(topicmodels)) (tmodel.lda <- LDA(td, control = list(alpha = 0.1), k = 5))
```

directory *Function to declare a connection to a directory (containing files)*

Description

Function to declare a connection to a directory, although unlike file it does not require closing. If the directory does not exist, the function will return an error.

Usage

```
directory(path = NULL)
```

Arguments

path	String describing the full path of the directory or NULL to use a GUI to choose a directory from disk

Examples

```
## Not run:
# name a directory of files
mydir <- directory("~/Dropbox/QUANTESS/corpora/ukManRenamed")
corpus(mydir)

# choose a directory using a GUI
corpus(directory())
## End(Not run)
```

docnames *extract document names*

Description

Extract the document names from a corpus or a document-feature matrix. Document names are the rownames of the documents data.frame in a corpus, or the rownames of the dfm object for a dfm. of the dfm object.

`docnames` queries the document names of a corpus or a dfm

`docnames <-` assigns new values to the document names of a corpus. (Does not work for dfm objects, whose document names are fixed,)

Usage

```
docnames(x)

## S3 method for class corpus
docnames(x)

docnames(x) <- value

## S3 method for class dfm
docnames(x)
```

Value

docnames returns a character vector of the document names

docnames<- assigns a character vector of the document names in a corpus

Examples

```
# query the document names of the inaugural speech corpus
docnames(inaugCorpus) <- paste("Speech", 1:ndoc(inaugCorpus), sep="")

# reassign the document names of the inaugural speech corpus
docnames(inaugCorpus) <- paste("Speech", 1:ndoc(inaugCorpus), sep="")
#
# query the document names of a dfm
docnames(dfm(inaugTexts[1:5]))
```

---

docvars *get or set for document-level variables*

Description

Get or set variables for the documents in a corpus

Usage

```
docvars(x)

docvars(x, field) <- value
```

Arguments

x	corpus whose document-level variables will be read or set
field	string containing the document-level variable name

Value

docvars returns a data.frame of the document-level variables

docvars<- assigns value to the named field

Examples

```
head(docvars(inaugCorpus))
docvars(inaugCorpus, "President") <- paste("prez", 1:ndoc(inaugCorpus), sep="")
head(docvars(inaugCorpus))
```

features.dfm *extract the feature labels from a dfm*

Description

Extract the features from a document-feature matrix, which are stored as the column names of the dfm object.

Usage

```
## S3 method for class dfm
features(x)
```

Arguments

x	the object (dfm) whose features will be extracted

Value

Character vector of the features

Examples

```
features(dfm(inaugTexts))[1:50]  # first 50 features (alphabetically sorted)
```

flatten.dictionary *Flatten a hierarchical dictionary into a list of character vectors*

Description

Converts a hierarchical dictionary (a named list of named lists, ending in character vectors at the lowest level) into a flat list of character vectors. Works like `unlist(dictionary, recursive=TRUE)` except that the recursion does not go to the bottom level.

Usage

```
flatten.dictionary(elms, parent = "", dict = list())
```

Arguments

elms	list to be flattened
parent	parent list name, gets built up through recursion in the same way that `unlist(dictionary, recursi` works
dict	the bottom list of dictionary entries ("synonyms") passed up from recursive calls

Details

Called by dfm()

Value

A dictionary flattened down one level further than the one passed

Author(s)

Kohei Watanabe

Examples

```
dictPopulismEN <-
    list(populism=c("elit*", "consensus*", "undemocratic*", "referend*",
                    "corrupt*", "propagand", "politici*", "*deceit*",
                    "*deceiv*", "*betray*", "shame*", "scandal*", "truth*",
                    "dishonest*", "establishm*", "ruling*"))
flatten.dictionary(dictPopulismEN)

hdict <- list(level1a = list(level1a1 = c("l1a11", "l1a12"),
                             level1a2 = c("l1a21", "l1a22")),
              level1b = list(level1b1 = c("l1b11", "l1b12"),
                             level1b2 = c("l1b21", "l1b22", "l1b23")),
              level1c = list(level1c1a = list(level1c1a1 = c("lowest1", "lowest2")),
                             level1c1b = list(level1c1b1 = c("lowestalone"))))
flatten.dictionary(hdict)
```

---

getRootFileNames *Truncate absolute filepaths to root filenames*

Description

This function takes an absolute filepath and returns just the document name

Usage

```
getRootFileNames(longFilenames)
```

Arguments

longFilenames	Absolute filenames including a full path with directory

Value

character vector of filenames withouth directory path

Author(s)

Paul Nulty

Examples

```
## Not run:
getRootFileNames(/home/paul/documents/libdem09.txt)

## End(Not run)
```

getTextDir *loads all text files from a given directory*

Description

given a directory name, get a list of all files in that directory and load them into a character vector using getTextFiles

Usage

```
getTextDir(dirname, enc = "detect", pattern = "\\.txt$")
```

Arguments

dirname	A directory path

Value

character vector of texts read from disk

Author(s)

Paul Nulty

Examples

```
## Not run:
getTextDir(/home/paul/documents/)

## End(Not run)
```

getTextDirGui *provides a gui interface to choose a gui to load texts from*

Description

launches a GUI to allow the user to choose a directory from which to load all files.

Usage

```
getTextDirGui()
```

Value

character vector of texts read from disk

Author(s)

Paul Nulty

Examples

```
## Not run:
getTextFiles(/home/paul/documents/libdem09.txt)

## End(Not run)
```

`getTextFiles` *load text files from disk into a vector of character vectors points to files, reads them into a character vector of the texts with optional names, default being filenames returns a named vector of complete, unedited texts*

Description

load text files from disk into a vector of character vectors points to files, reads them into a character vector of the texts with optional names, default being filenames returns a named vector of complete, unedited texts

Usage

```
getTextFiles(filenames, textnames = NULL, enc = "unknown",
  verbose = FALSE)
```

Arguments

`filenames`	a vector of paths to text files
`textnames`	names to assign to the texts
`verbose`	If TRUE, print out names of files being read. Default is FALSE

Value

character vector of texts read from disk

Author(s)

Paul Nulty

Examples

```
## Not run:
getTextFiles(/home/paul/documents/libdem09.txt)

## End(Not run)
```

inaugCorpus *A corpus of US presidential inaugural addresses from 1789-2013*

Description

inaugCorpus is the quanteda corpus object of US presidents' inaugural addresses since 1789. Document variables contain the year of the address and the last name of the president.

inaugTexts is the character vector of US presidential inaugaration speeches

References

https://archive.org/details/Inaugural-Address-Corpus-1789-2009 and http://www.presidency.ucsb.edu/inaugurals.php.

Examples

```
# some operations on the inaugural corpus
data(inaugCorpus)
summary(inaugCorpus)
head(docvars(inaugCorpus), 10)
# working with the character vector only
data(inaugTexts)
str(inaugTexts)
head(docvars(inaugCorpus), 10)
mycorpus <- corpus(inaugTexts)
```

kwic *List key words in context from a text or a corpus of texts.*

Description

For a text or a collection of texts (in a quanteda corpus object), return a list of a keyword supplied by the user in its immediate context, identifying the source text and the word index number within the source text. (Not the line number, since the text may or may not be segmented using end-of-line delimiters.)

Usage

```
kwic(x, word, window = 5, regex = TRUE)

## S3 method for class character
kwic(x, word, window = 5, regex = TRUE)

## S3 method for class corpus
kwic(x, word, window = 5, regex = TRUE)
```

Arguments

x	A text character scalar or a quanteda corpus. (Currently does not support character vectors.)
word	A keyword chosen by the user.
window	The number of context words to be displayed around the keyword.
regex	If TRUE (default), then "word" is a regular expression, otherwise only match the whole word. Note that if regex=TRUE and no special regular expression characters are used in the search query, then the concordance will include all words in which the search term appears, and not just when it appears as an entire word. (For instance, searching for the word "key" will also return "whiskey".)

Value

A data frame with the context before (preword), the keyword in its original format (word, preserving case and attached punctuation), and the context after (postword). The rows of the dataframe will be named with the word index position, or the text name and the index position for a corpus object.

Author(s)

Kenneth Benoit and Paul Nulty

Examples

```
kwic(inaugTexts, "terror")
kwic(inaugTexts, "terror", regex=FALSE)  # returns only whole word, without trailing punctuation
```

---

language *get or set the language of corpus documents*

Description

Get or set the _language document-level metadata field in a corpus. Same as

Usage

```
language(corp)
```

---

likelihood.test *likelihood test for contingency tables*

Description

returns a list of values

Usage

```
likelihood.test(x)
```

Arguments

x	a contingency table or matrix object

Value

A list of return values

Author(s)

Kenneth Benoit

---

metacorpus *get or set corpus metadata*

Description

Get or set the corpus-level metadata in a quanteda corpus object.

Usage

```
metacorpus(corp, field = NULL)

metacorpus(corp, field) <- value
```

Arguments

corp	A quanteda corpus object
field	Metadata field name(s). If NULL (default), return all metadata names.

Value

For metacorpus, a list of the metadata fields in the corpus. If a list is not what you wanted, you can wrap the results in unlist, but this will remove any metadata field that is set to NULL.

For metacorpus <-, the corpus with the updated metadata.

Examples

```
metacorpus(inaugCorpus)
metacorpus(inaugCorpus, "source")
metacorpus(inaugCorpus, "citation") <- "Presidential Speeches Online Project (2014)."
metacorpus(inaugCorpus, "citation")
```

metadoc *get or set document-level meta-data*

Description

Get or set the document-level meta-data, including reserved fields for language and corpus.

Usage

```
metadoc(corp, field = NULL)
```

Arguments

corp	A quanteda corpus object

Value

For `texts`, a character vector of the texts in the corpus.

For `texts <-`, the corpus with the updated texts.

Note

Document-level meta-data names are preceded by an underscore character, such as `_encoding`, but when named in in the `field` argument, do *not* need the underscore character.

Examples

```
mycorp <- subset(inaugCorpus, Year>1990)
summary(mycorp, showmeta=TRUE)
metadoc(mycorp, "encoding") <- "UTF-8"
metadoc(mycorp)
metadoc(mycorp, "language") <- "english"
summary(mycorp, showmeta=TRUE)
```

ndoc *get the number of documents*

Description

Returns the number of documents in a corpus objects

Usage

```
## S3 method for class corpus
ndoc(x, ...)

## S3 method for class dfm
ndoc(x, ...)
```

Arguments

x	a corpus or dfm object
...	additional parameters

Value

an integer (count) of the number of documents in the corpus or dfm

Examples

```
ndoc(inaugCorpus)
ndoc(dfm(inaugCorpus))
```

---

ngrams *Create ngrams*

Description

Create a set of ngrams (words in sequence) from text(s) in a character vector

Usage

```
ngrams(text, n = 2, concatenator = "_", include.all = FALSE, ...)
```

Arguments

text	character vector containing the texts from which ngrams will be extracted
n	the number of tokens to concatenate. Default is 2 for bigrams.
concatenator	character for combining words, default is _ (underscore) character
include.all	if TRUE, add n-1...1 grams to the returned list
...	additional parameters

Details

... provides additional arguments passed to tokenize

Value

a list of character vectors of ngrams, one list element per text

Author(s)

Ken Benoit, Kohei Watanabe, Paul Nulty

Examples

```
ngrams("The quick brown fox jumped over the lazy dog.", n=2)
identical(ngrams("The quick brown fox jumped over the lazy dog.", n=2),
          bigrams("The quick brown fox jumped over the lazy dog.", n=2))
ngrams("The quick brown fox jumped over the lazy dog.", n=3)
ngrams("The quick brown fox jumped over the lazy dog.", n=3, concatenator="~")
ngrams("The quick brown fox jumped over the lazy dog.", n=3, include.all=TRUE)
```

plot.dfm *plot features as a wordcloud*

Description

The default plot method for a dfm object. Produces a wordcloud plot for the features of the dfm, weighted by the total frequencies. To produce word cloud plots for specific documents, the only way currently to do this is to produce a dfm only from the documents whose features you want plotted.

Usage

```
## S3 method for class dfm
plot(x, ...)
```

Arguments

x	a dfm object

Details

... provides additional parameters passed to to wordcloud or to text (and strheight, strwidth)

See Also

wordcloud

Examples

```
# plot the features (without stopwords) from Obamas two inaugural addresses
mydfm <- dfm(subset(inaugCorpus, President=="Obama"), verbose=FALSE, stopwords=TRUE)
plot(mydfm)

# plot only Lincolns inaugural address
plot(dfm(subset(inaugCorpus, President=="Lincoln"), verbose=FALSE, stopwords=TRUE))

# plot in colors with some additional options passed to wordcloud
plot(mydfm, random.color=TRUE, rot.per=.25, colors=sample(colors()[2:128], 5))
```

quanteda *An R package for the quantitative analysis of textual data.*

Description

A set of functions for creating and managing text corpora, extracting features from text corpora, and analyzing those features using quantitative methods.

Author(s)

Ken Benoit and Paul Nulty

readWStatDict *Import a Wordstat dictionary*

Description

Make a flattened list from a hierarchical wordstat dictionary

Usage

```
readWStatDict(path)
```

Arguments

path	path to the wordstat dictionary file (.cat)

Value

a named list, where each the name of element is a bottom level category in the hierarchical wordstat dictionary. Each element is a list of the dictionary terms corresponding to that level.

Author(s)

Kohei Watanabe

Examples

```
## Not run:
path <- ~/Dropbox/QUANTESS/corpora/LaverGarry.cat
lgdict <- readWStatDict(path)

## End(Not run)
```

segment *segment texts into component elements*

Description

Segment text(s) into tokens, sentences, paragraphs, or other sections. segment works on a character vector or corpus object, and allows the delimiters to be defined. See details.

Usage

```
segment(x, ...)

## S3 method for class character
segment(x, what = c("tokens", "sentences", "paragraphs",
  "other"), delimiter = ifelse(what == "tokens", " ", ifelse(what ==
  "sentences", "[.!?:;]", "\\n{2}")), ...)

## S3 method for class corpus
segment(x, what = c("tokens", "sentences", "paragraphs",
  "other"), delimiter = ifelse(what == "tokens", " ", ifelse(what ==
  "sentences", "[.!?:;]", "\\n{2}")), ...)
```

Arguments

x	defines the component to define the segmentation unit. Current options are tokens, sentences, paragraphs, and other. Segmenting on other allows segmentation of a text on any user-defined value, and must be accompanied by the delimiter argument.
...	provides additional arguments to be passed to clean
what	unit of segmentation
delimiter	delimiter defined as a regex for segmentation. Each type has its own default, except other, which requires a value to be specified.

Details

Tokens are delimited by whitespace. For sentences, the delimiter can be defined by the user. The default for sentences includes ., !, ?, plus ; and :.

For paragraphs, the default is two carriage returns, although this could be changed to a single carriage return by changing the value of delimiter to "\\n{1}" which is the R version of the regex for one newline character. (You might need this if the document was created in a word processor, for instance, and the lines were wrapped in the window rather than being hard-wrapped with a newline character.)

Value

A list of segmented texts, with each element of the list correponding to one of the original texts.

Examples

```
# same as tokenize()
identical(tokenize(uk2010immig, lower=FALSE), segment(uk2010immig, lower=FALSE))

# segment into paragraphs
segment(uk2010immig[3:4], "paragraphs")

# segment a text into sentences
segmentedChar <- segment(uk2010immig, "sentences")
segmentedChar[2]
# segment a corpus into sentences
segmentedCorpus <- segment(corpus(uk2010immig), "sentences")
identical(segmentedCorpus, segmentedChar)
```

---

settings *Get or set the corpus settings*

Description

Get or set the corpus settings

Get or set various settings in the corpus for the treatment of texts, such as rules for stemming, stopwords, collocations, etc. settings(corp) query the corps settings settings(corp, settingname) <- update the corpus settings

Get the settings from a which a dfm was created

Usage

```
settings(x, ...)

## S3 method for class corpus
settings(x, fields = NULL, ...)

settings(corp, fields) <- value

## S3 method for class dfm
settings(x, ...)
```

Arguments

x	dfm from which settings are queried
fields	a valid corpus setting field name
corp	Corpus from/to which settings are queried or applied

Examples

```
settings(inaugCorpus, "stopwords")
tempdfm <- dfm(inaugCorpus)
tempdfmSW <- dfm(inaugCorpus, stopwords=TRUE)
settings(inaugCorpus, "stopwords") <- TRUE
tempdfmSW <- dfm(inaugCorpus)
tempdfm <- dfm(inaugCorpus, stem=TRUE)
settings(tempdfm)
```

---

sort.dfm *sort a dfm by one or more margins*

Description

Sorts a dfm by frequency of total features, total features in documents, or both

Usage

```
## S3 method for class dfm
sort(x, decreasing = TRUE, margin = c("features", "docs",
  "both"), ...)
```

Arguments

x	Document-feature matrix created by dfm
decreasing	TRUE (default) if sort will be in descending order, otherwise sort in increasing order
margin	which margin to sort on features to sort by frequency of features, docs to sort by total feature counts in documents, and both to sort by both

Value

A sorted dfm matrix object

Author(s)

Ken Benoit

Examples

```
dtm <- dfm(inaugCorpus)
dtm[1:10, 1:5]
dtm <- sort(dtm)
sort(dtm)[1:10, 1:5]
sort(dtm, TRUE, "both")[1:10, 1:5]  # note that the decreasing=TRUE argument
                                    # must be second, because of the order of the
                                    # formals in the generic method of sort()
```

---

stopwords *A named list containing common stopwords in 14 languages*

Description

SMART English stopwords from the SMART information retrieval system (obtained from http://jmlr.csail.mit.edu/papers smart-stop-list/english.stop) and a set of stopword lists from the Snowball stemmer project in different languages (obtained from http://svn.tartarus.org/snowball/trunk/website/algorithms/*/stop.txt). Supported languages are danish, dutch, english, finnish, french, german, hungarian, italian, norwegian, portuguese, russian, spanish, and swedish. Language names are case sensitive. Alternatively, their IETF language tags may be used.

---

stopwordsGet *access stopwords*

Description

This function retrieves stopwords from the type specified in the `kind` argument and returns the stopword list as a character vector The default is English. See stopwords for information about the list.

Usage

```
stopwordsGet(kind = "english")
```

Arguments

`kind`	The pre-set kind of stopwords (as a character string)

Value

a character vector or dfm with stopwords removed

Examples

```
stopwordsGet()
stopwordsGet("italian")
```

stopwordsRemove *remove stopwords from a text or dfm*

Description

This function takes a character vector or dfm and removes words in the remove common or 'semantically empty' words from a text. See stopwordsGet for the information about the default lists.

Usage

```
stopwordsRemove(text, stopwords = NULL)

## S3 method for class character
stopwordsRemove(text, stopwords = NULL)

## S3 method for class dfm
stopwordsRemove(text, stopwords = NULL)
```

Arguments

text	Text from which stopwords will be removed
stopwords	Character vector of stopwords to remove - if none is supplied, a default set of English stopwords is used

Details

This function takes a character vector 'text' and removes words in the list provided in stopwords. If no list of stopwords is provided a default list for English is used. The function stopwordsGet can load a default set of stopwords for many languages.

Value

a character vector or dfm with stopwords removed

Examples

```
## examples for character objects
someText <- "Here is an example of text containing some stopwords we want to remove."
itText <- "Ecco un esempio di testo contenente alcune parole non significative che vogliamo rimuovere."
stopwordsRemove(someText)
stopwordsRemove(someText, stopwordsGet("SMART"))
stopwordsRemove(itText, stopwordsGet("italian"))
stopwordsRemove(someText, c("containing", "example"))

## example for dfm objects
docmat <- dfm(uk2010immig)
docmatNostopwords <- stopwordsRemove(docmat)
dim(docmat)
dim(docmatNostopwords)
dim(stopwordsRemove(docmat, stopwordsGet("SMART")))
```

subset.corpus *extract a subset of a corpus*

Description

Works just like the normal subset command but for corpus objects

Usage

```
## S3 method for class corpus
subset(x, subset = NULL, select = NULL, ...)
```

Arguments

x	corpus object to be subsetted.
subset	logical expression indicating elements or rows to keep: missing values are taken as false.
select	expression, indicating the attributes to select from the corpus

Value

corpus object

Examples

```
## Not run:
data(inaugCorpus)
summary(subset(inaugCorpus, Year>1980))

## End(Not run)
```

summary.corpus *Corpus summary*

Description

Displays information about a corpus object, including attributes and metadata such as date of number of texts, creation and source.

Usage

```
## S3 method for class corpus
summary(object, n = 100, verbose = TRUE,
  showmeta = FALSE, ...)
```

Arguments

object	corpus to be summarized
n	maximum number of texts to describe, default=100
verbose	FALSE to turn off printed output
showmeta	TRUE to include document-level meta-data

Examples

```
summary(inaugCorpus)
summary(inaugCorpus, n=10)
mycorpus <- corpus(uk2010immig, docvars=data.frame(party=names(uk2010immig)), enc="UTF-8")
summary(mycorpus, showmeta=TRUE)  # show the meta-data
mysummary <- summary(mycorpus, verbose=FALSE)  # (quietly) assign the results
mysummary$Types / mysummary$Tokens             # crude type-token ratio
```

---

syllableCounts *A named list mapping words to counts of their syllables*

Description

A named list mapping words to counts of their syllables, generated from the CMU pronunciation dictionary

References

http://www.speech.cs.cmu.edu/cgi-bin/cmudict

Examples

```
data(syllableCounts)
syllableCounts["sixths"]
syllableCounts["onomatopeia"]
```

---

texts *get or set corpus texts*

Description

Get or replace the texts in a quanteda corpus object.

Usage

```
texts(corp)

texts(corp) <- value
```

Arguments

corp	A quanteda corpus object
rownames	If TRUE, overwrite the names of the documents with names from assigned object.

Value

For texts, a character vector of the texts in the corpus.

For texts <-, the corpus with the updated texts.

Examples

```
texts(inaugCorpus)[1]
sapply(texts(inaugCorpus), nchar)  # length in characters of the inaugual corpus texts

## this doesnt work yet - need to overload [ for this replacement function
# texts(inaugTexts)[55] <- "GW Bushs second inaugural address, the condensed version."
```

---

tf *normalizes the term frequencies a dfm*

Description

Returns a matrix of term weights, as a dfm object

Usage

```
tf(x)
```

Arguments

dfm	Document-feature matrix created by dfm

Value

A dfm matrix object where values are relative term proportions within the document

Author(s)

Ken Benoit

Examples

```
data(inaugCorpus)
dtm <- dfm(inaugCorpus)
dtm[1:10, 100:110]
tf(dtm)[1:10, 100:110]
```

---

tfidf.dfm *compute the tf-idf weights of a dfm*

Description

Returns a matrix of tf-idf weights, as a dfm object

Usage

```
## S3 method for class dfm
tfidf(x, normalize = TRUE, ...)
```

Arguments

x	document-feature matrix created by dfm
normalize	whether to normalize term frequency by document totals

Value

A dfm matrix object where values are tf-idf weights

Author(s)

Ken Benoit

Examples

```
data(inaugCorpus)
dtm <- dfm(inaugCorpus)
dtm[1:10, 100:110]
tfidf(dtm)[1:10, 100:110]
tfidf(dtm, normalize=FALSE)[1:10, 100:110]
```

---

tokenize *tokenize a set of texts*

Description

Tokenize the texts from a character vector or from a corpus.

Usage

```
tokenize(x, ...)

## S3 method for class character
tokenize(x, simplify = FALSE, sep = " ", ...)

## S3 method for class corpus
tokenize(x, ...)
```

Arguments

x	The text(s) or corpus to be tokenized
simplify	If TRUE, return a character vector of tokens rather than a list of length ndoc(texts), with each element of the list containing a character vector of the tokens corresponding to that text.
sep	by default, tokenize expects a 'white-space' delimiter between tokens. Alternatively, sep can be used to specify another character which delimits fields.

Details

... provides additional arguments passed to clean

Value

A list of length ndoc(x) of the tokens found in each text.

A list of length ndoc(texts) of the tokens found in each text.

Examples

```
# same for character vectors and for lists
tokensFromChar <- tokenize(inaugTexts)
tokensFromCorp <- tokenize(inaugCorpus)
identical(tokensFromChar, tokensFromCorp)
str(tokensFromChar)
# returned as a list
head(tokenize(inaugTexts[57])[[1]], 10)
# returned as a character vector using simplify=TRUE
head(tokenize(inaugTexts[57], simplify=TRUE), 10)

# demonstrate some options with clean
head(tokenize(inaugTexts[57], simplify=TRUE, lower=FALSE), 30)
```

---

topfeatures *list the most frequent features*

Description

List the most frequently occuring features.

Usage

```
topfeatures(x, n = 10, decreasing = TRUE)

## S3 method for class dfm
topfeatures(x, n = 10, decreasing = TRUE, ...)
```

Value

A named numeric vector of feature counts, where the names are the feature labels.

Examples

```
topfeatures(dfm(inaugCorpus))
topfeatures(dfm(inaugCorpus, stopwords=TRUE))
# least frequent features
topfeatures(dfm(inaugCorpus), decreasing=FALSE)
```

trimdfm *Trim a dfm based on a subset of features and words*

Description

Returns a document by feature matrix reduced in size based on document and term frequency, and/or subsampling.

Usage

```
trimdfm(x, minCount = 5, minDoc = 5, sample = NULL, verbose = TRUE)
```

Arguments

x	document-feature matrix created by dfm
minCount	minimum feature count
minDoc	minimum number of documents in which a feature appears
sample	how many features to retain (based on random selection)
verbose	print messages

Value

A dfm object reduced in size.

Author(s)

Ken Benoit adapted from code by Will Lowe (see trim)

Examples

```
data(inaugCorpus)
dtm <- dfm(inaugCorpus)
dim(dtm)
dtmReduced <- trimdfm(dtm, minCount=10, minDoc=2) # only words occuring at least 5 times and in at least 2
dim(dtmReduced)
dtmSampled <- trimdfm(dtm, sample=200)  # top 200 words
dim(dtmSampled)  # 196 x 200 words
```

twitterSearch *work-in-progress from-scratch interface to Twitter search API*

Description

work-in-progress from-scratch interface to Twitter search API

Usage

```
twitterSearch()
```

twitterStreamer *work-in-progress interface to Twitter streaming API*

Description

work-in-progress interface to Twitter streaming API

Usage

```
twitterStreamer()
```

twitterTerms *make a corpus object from results of a twitter REST search*

Description

All of the attributes returned by the twitteR library call are included as attributes in the corpus. A oauth key is required, for further instruction about the oauth processs see: https://dev.twitter.com/apps/new and the twitteR documentation

Usage

```
twitterTerms(query, numResults = 50, key, cons_secret, token, access_secret)
```

Arguments

query	Search string for twitter
numResults	Number of results desired.
key	Number of results desired.
cons_secret	'your consumer secret here'
token	'your access token here'
access_secret	'your access secret here'
key	'your consumer key here'

Examples

```
## Not run:
twCorp <- twitterTerms(example, 10, key, cons_secret, token, access_secret)

## End(Not run)
```

uk2010immig *Immigration-related sections of 2010 UK party manifestos*

Description

Extracts from the election manifestos of 9 UK political parties from 2010, related to immigration or asylum-seekers.

Format

A named character vector of plain ASCII texts

Examples

```
data(uk2010immig)
uk2010immigCorpus <- corpus(uk2010immig, docvars=list(party=names(uk2010immig)))
language(uk2010immigCorpus) <- "english"
encoding(uk2010immigCorpus) <- "UTF-8"
summary(uk2010immigCorpus)
```

Index